COGNIFYZ TASK_1

1 Predict_Restaurant_Ratings_Cognifyz_Task1

1.0.1 Build a machine learning model to predict the aggregate rating of a restaurant based on other features.

[1]:
```
# Importing the Libraries
import pandas as pd
import numpy as np
```

[2]:
```
import warnings
warnings.filterwarnings("ignore")
```

1.1 Pre-Processing Steps

[5]:
```
# Creating the Dataframe
file_path = r'D:\Software\New Project\Internship\Cognifyz\Predict Restaurant␣
 ↪Ratings\Dataset .csv'
df = pd.read_csv(file_path)
df.head()
```

[5]:

	Restaurant ID	Restaurant Name	Country Code	City	\
0	6317637	Le Petit Souffle	162	Makati City	
1	6304287	Izakaya Kikufuji	162	Makati City	
2	6300002	Heat - Edsa Shangri-La	162	Mandaluyong City	
3	6318506	Ooma	162	Mandaluyong City	
4	6314302	Sambo Kojin	162	Mandaluyong City	

	Address	\
0	Third Floor, Century City Mall, Kalayaan Avenu…	
1	Little Tokyo, 2277 Chino Roces Avenue, Legaspi…	
2	Edsa Shangri-La, 1 Garden Way, Ortigas, Mandal…	
3	Third Floor, Mega Fashion Hall, SM Megamall, O…	
4	Third Floor, Mega Atrium, SM Megamall, Ortigas…	

	Locality	\
0	Century City Mall, Poblacion, Makati City	
1	Little Tokyo, Legaspi Village, Makati City	

```
2  Edsa Shangri-La, Ortigas, Mandaluyong City
3      SM Megamall, Ortigas, Mandaluyong City
4      SM Megamall, Ortigas, Mandaluyong City

                                    Locality Verbose   Longitude   Latitude  \
0  Century City Mall, Poblacion, Makati City, Mak…  121.027535  14.565443
1  Little Tokyo, Legaspi Village, Makati City, Ma…  121.014101  14.553708
2  Edsa Shangri-La, Ortigas, Mandaluyong City, Ma…  121.056831  14.581404
3  SM Megamall, Ortigas, Mandaluyong City, Mandal…  121.056475  14.585318
4  SM Megamall, Ortigas, Mandaluyong City, Mandal…  121.057508  14.584450

                           Cuisines  …          Currency Has Table booking  \
0       French, Japanese, Desserts  …  Botswana Pula(P)               Yes
1                         Japanese  …  Botswana Pula(P)               Yes
2  Seafood, Asian, Filipino, Indian  …  Botswana Pula(P)               Yes
3                  Japanese, Sushi  …  Botswana Pula(P)                No
4                 Japanese, Korean  …  Botswana Pula(P)               Yes

  Has Online delivery Is delivering now Switch to order menu Price range  \
0                  No                 No                    No           3
1                  No                 No                    No           3
2                  No                 No                    No           4
3                  No                 No                    No           4
4                  No                 No                    No           4

   Aggregate rating  Rating color Rating text Votes
0               4.8    Dark Green   Excellent   314
1               4.5    Dark Green   Excellent   591
2               4.4         Green   Very Good   270
3               4.9    Dark Green   Excellent   365
4               4.8    Dark Green   Excellent   229

[5 rows x 21 columns]
```

[ ]:
```
df = df.drop('Restaurant ID', axis=1)
df = df.drop('Restaurant Name', axis=1)
df = df.drop('Country Code', axis=1)
df = df.drop('City', axis=1)
df = df.drop('Address', axis=1)
df = df.drop('Locality', axis=1)
df = df.drop('Locality Verbose', axis=1)
df = df.drop('Longitude', axis=1)
df = df.drop('Latitude', axis=1)
df = df.drop('Cuisines', axis=1)
df = df.drop('Currency', axis=1)
```

[8]:
```
df
```

[8]:
```
      Average Cost for two Has Table booking Has Online delivery  \
0                     1100               Yes                   No
1                     1200               Yes                   No
2                     4000               Yes                   No
3                     1500                No                   No
4                     1500               Yes                   No
…                      …                  …                    …
9546                    80                No                   No
9547                   105                No                   No
9548                   170                No                   No
9549                   120                No                   No
9550                    55                No                   No

     Is delivering now Switch to order menu  Price range  Aggregate rating  \
0                   No                   No            3               4.8
1                   No                   No            3               4.5
2                   No                   No            4               4.4
3                   No                   No            4               4.9
4                   No                   No            4               4.8
…                  …                    …            …               …
9546                No                   No            3               4.1
9547                No                   No            3               4.2
9548                No                   No            4               3.7
9549                No                   No            4               4.0
9550                No                   No            2               4.0

     Rating color Rating text  Votes
0      Dark Green   Excellent    314
1      Dark Green   Excellent    591
2           Green   Very Good    270
3      Dark Green   Excellent    365
4      Dark Green   Excellent    229
…           …         …    …
9546        Green   Very Good    788
9547        Green   Very Good   1034
9548       Yellow        Good    661
9549        Green   Very Good    901
9550        Green   Very Good    591

[9551 rows x 10 columns]
```

[9]:
```
df.shape
```

[9]:
```
(9551, 10)
```

[10]:
```
df.info
```

[10]:
```
<bound method DataFrame.info of       Average Cost for two Has Table booking Has
Online delivery  \
0                        1100               Yes                  No
1                        1200               Yes                  No
2                        4000               Yes                  No
3                        1500                No                  No
4                        1500               Yes                  No
…                         …                  …                   …
9546                       80                No                  No
9547                      105                No                  No
9548                      170                No                  No
9549                      120                No                  No
9550                       55                No                  No

     Is delivering now Switch to order menu  Price range  Aggregate rating  \
0                   No                    No            3               4.8
1                   No                    No            3               4.5
2                   No                    No            4               4.4
3                   No                    No            4               4.9
4                   No                    No            4               4.8
…                  …                     …           …                …
9546                No                    No            3               4.1
9547                No                    No            3               4.2
9548                No                    No            4               3.7
9549                No                    No            4               4.0
9550                No                    No            2               4.0

     Rating color Rating text  Votes
0      Dark Green   Excellent    314
1      Dark Green   Excellent    591
2           Green   Very Good    270
3      Dark Green   Excellent    365
4      Dark Green   Excellent    229
…           …         …    …
9546        Green   Very Good    788
9547        Green   Very Good   1034
9548       Yellow        Good    661
9549        Green   Very Good    901
9550        Green   Very Good    591

[9551 rows x 10 columns]>
```

[11]:
```
df.describe()
```

[11]:
```
       Average Cost for two  Price range  Aggregate rating         Votes
count           9551.000000  9551.000000       9551.000000   9551.000000
mean            1199.210763     1.804837          2.666370    156.909748
```

```
std            16121.183073     0.905609          1.516378    430.169145
min                0.000000     1.000000          0.000000      0.000000
25%              250.000000     1.000000          2.500000      5.000000
50%              400.000000     2.000000          3.200000     31.000000
75%              700.000000     2.000000          3.700000    131.000000
max           800000.000000     4.000000          4.900000  10934.000000
```

[12]:
```
# Checking for missing values
df.isnull().sum()
```

[12]:
```
Average Cost for two    0
Has Table booking       0
Has Online delivery     0
Is delivering now       0
Switch to order menu    0
Price range             0
Aggregate rating        0
Rating color            0
Rating text             0
Votes                   0
dtype: int64
```

[13]:
```
# Checking for duplicated values
df.duplicated().sum()
```

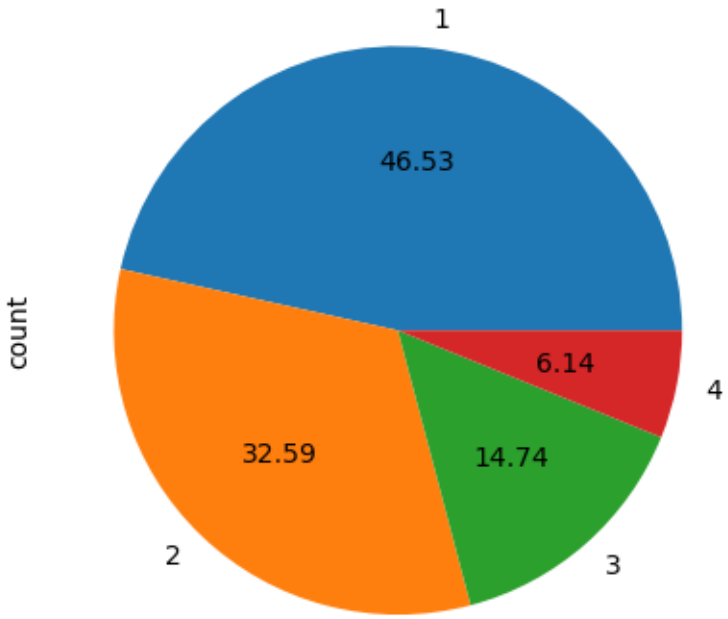
[13]:
```
2871
```

[14]:
```
df.dropna(inplace=True)
```

[15]:
```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

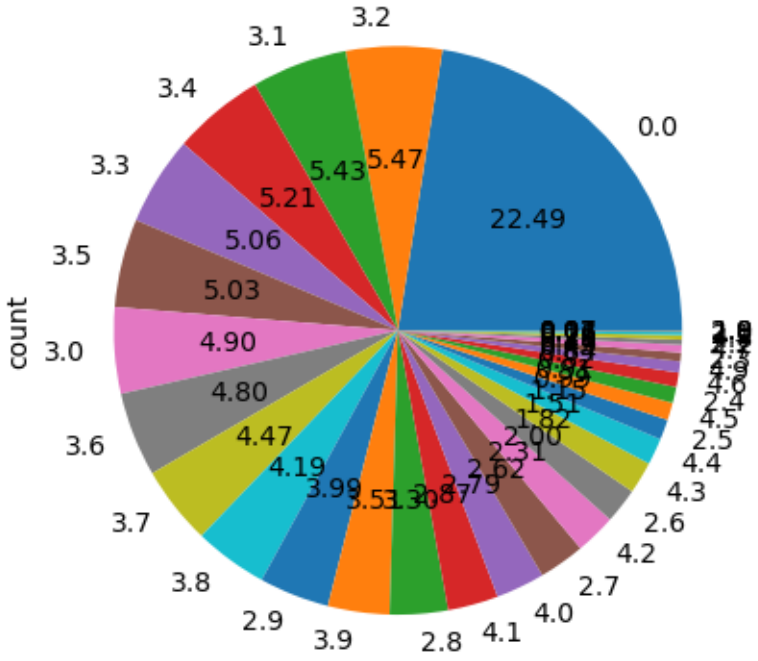
[16]:
```
df['Price range'].value_counts().plot(kind='pie', autopct = '%.2f')
```

[16]:
```
<Axes: ylabel='count'>
```



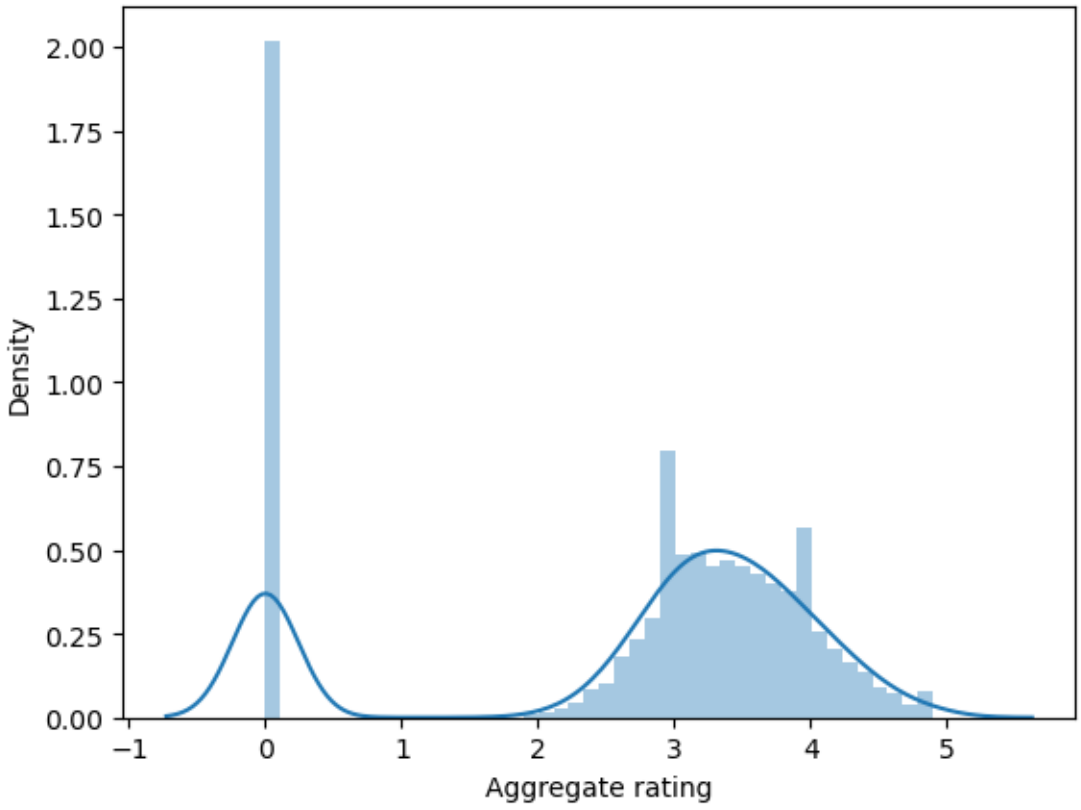
[17]:
```
df['Aggregate rating'].value_counts().plot(kind='pie', autopct = '%.2f')
```

[17]: <Axes: ylabel='count'>



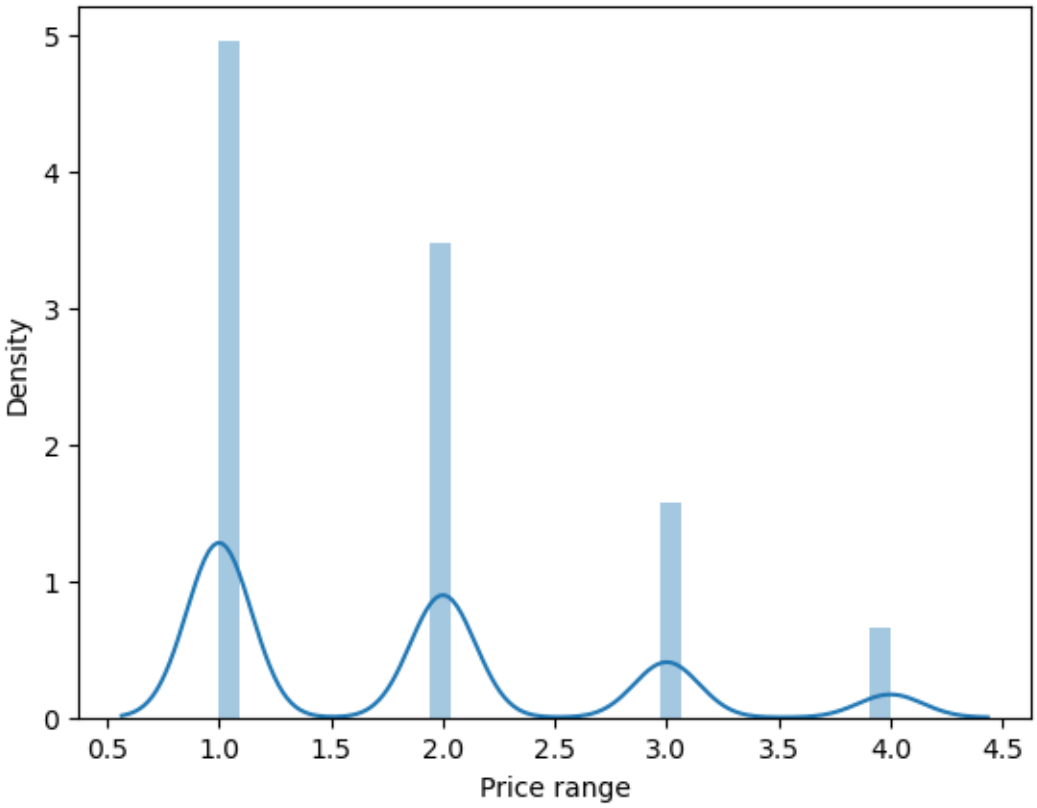
[18]:
```
sns.distplot(df['Aggregate rating'])
```

[18]: <Axes: xlabel='Aggregate rating', ylabel='Density'>



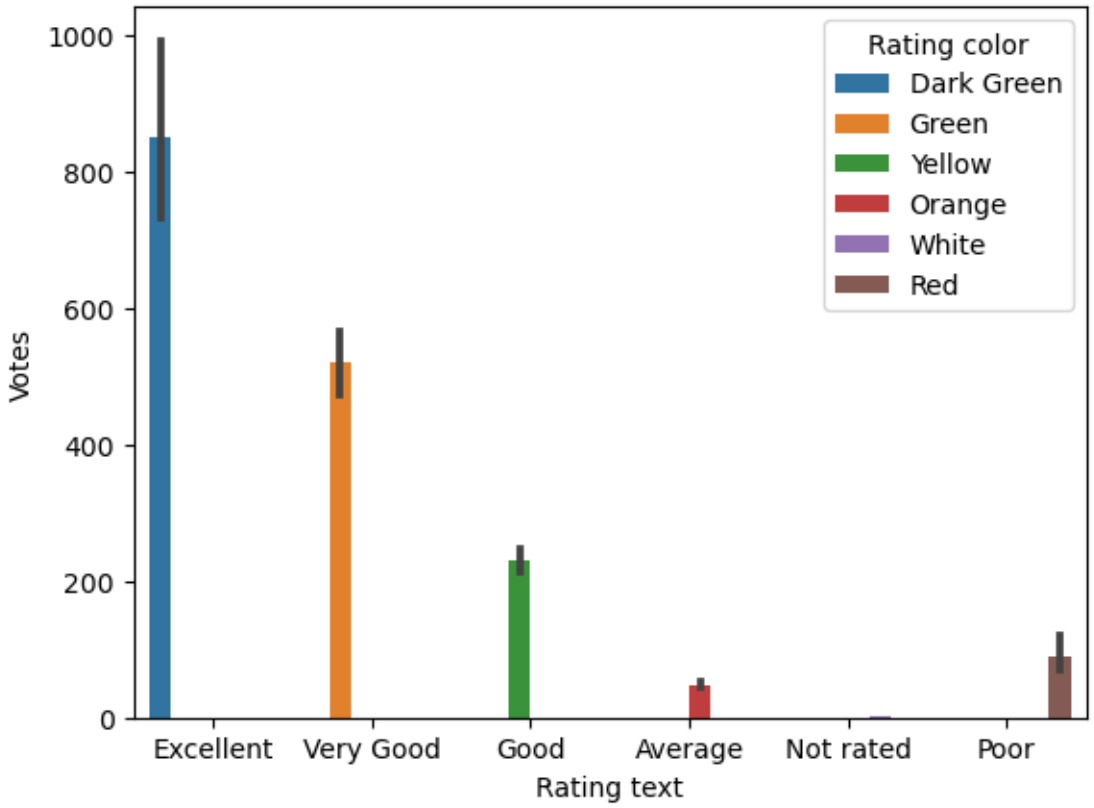
[19]:
```
sns.distplot(df['Price range'])
```

[19]:
```
<Axes: xlabel='Price range', ylabel='Density'>
```



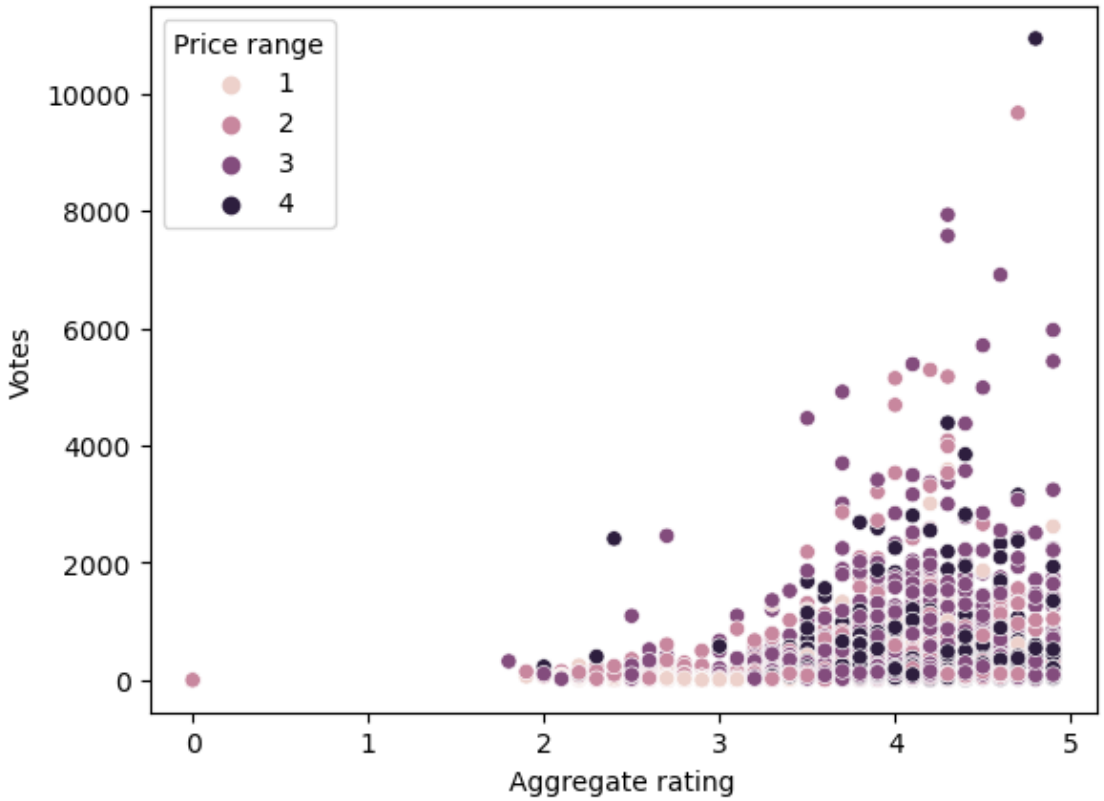
[20]:
```
sns.barplot(x=df["Rating text"],y=df["Votes"],hue =df["Rating color"])
```

[20]:
```
<Axes: xlabel='Rating text', ylabel='Votes'>
```



[21]:
```
sns.scatterplot(x=df["Aggregate rating"],y=df["Votes"],hue=df["Price range"])
```

[21]: <Axes: xlabel='Aggregate rating', ylabel='Votes'>



[22]:
```
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
df['Has Table booking'] = label_encoder.fit_transform(df['Has Table booking'])
df['Has Online delivery'] = label_encoder.fit_transform(df['Has Online␣
 ↪delivery'])
df['Is delivering now'] = label_encoder.fit_transform(df['Is delivering now'])
df['Switch to order menu'] = label_encoder.fit_transform(df['Switch to order␣
 ↪menu'])
df['Rating color'] = label_encoder.fit_transform(df['Rating color'])
df['Rating text'] = label_encoder.fit_transform(df['Rating text'])
```

[23]:
```
df
```

[23]:

	Average Cost for two	Has Table booking	Has Online delivery	\
0	1100	1	0	
1	1200	1	0	
2	4000	1	0	
3	1500	0	0	
4	1500	1	0	
…	…	…	…	
9546	80	0	0	

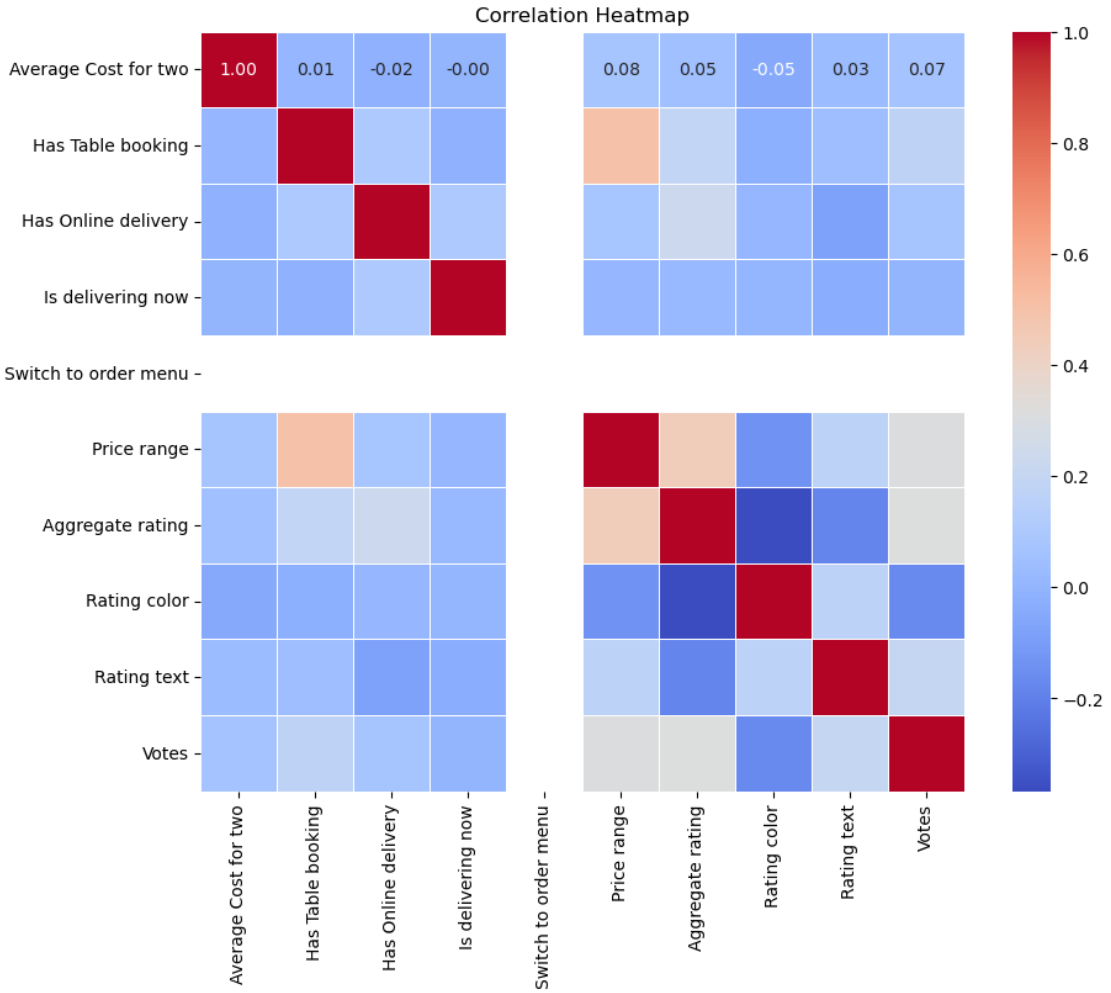
```
9547                  105                    0                     0
9548                  170                    0                     0
9549                  120                    0                     0
9550                   55                    0                     0

      Is delivering now  Switch to order menu  Price range  Aggregate rating  \
0                     0                      0            3               4.8
1                     0                      0            3               4.5
2                     0                      0            4               4.4
3                     0                      0            4               4.9
4                     0                      0            4               4.8
…                   …                      …          …               …
9546                  0                      0            3               4.1
9547                  0                      0            3               4.2
9548                  0                      0            4               3.7
9549                  0                      0            4               4.0
9550                  0                      0            2               4.0

      Rating color  Rating text  Votes
0                0            1    314
1                0            1    591
2                1            5    270
3                0            1    365
4                0            1    229
…              …          …    …
9546             1            5    788
9547             1            5   1034
9548             5            2    661
9549             1            5    901
9550             1            5    591

[9551 rows x 10 columns]
```

[24]:
```
correlation_matrix = df.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f",␣
 ↪linewidths=0.5)
plt.title('Correlation Heatmap')
plt.show()
```



[25]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import r2_score
```

[26]:
```
x = df.drop('Aggregate rating', axis=1)
y = df['Aggregate rating']
```

[27]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.
↪1,random_state=353)
x_train.head()
y_train.head()
```

[27]:
```
8696    0.0
1164    4.0
4824    3.2
7574    3.2
2961    2.3
Name: Aggregate rating, dtype: float64
```

1.2 Running the Linear Regression Model

[28]:
```
reg=LinearRegression()
reg.fit(x_train,y_train)
y_pred=reg.predict(x_test)
from sklearn.metrics import r2_score
r2_score(y_test,y_pred)
```

[28]:
```
0.44846419965192574
```

[29]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
reg = LinearRegression()
reg.fit(x_train, y_train)
y_pred = reg.predict(x_test)
mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error (MSE): {mse:.2f}")
r2 = r2_score(y_test, y_pred)
print(f"R-squared (R2) Error: {r2:.2f}")
```

```
Mean Squared Error (MSE): 1.36
R-squared (R2) Error: 0.45
```

1.3 Building the Decision Tree Regressor

[30]:
```
from sklearn.tree import DecisionTreeRegressor
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.
 ↪1,random_state=105)
DTree=DecisionTreeRegressor(min_samples_leaf=.0001)
DTree.fit(x_train,y_train)
y_predict=DTree.predict(x_test)
from sklearn.metrics import r2_score
r2_score(y_test,y_predict)
```

[30]:
```
0.9776218484219719
```

[31]:
```
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
```

```
DTree = DecisionTreeRegressor(min_samples_leaf=0.0001)

DTree.fit(x_train, y_train)

y_predict = DTree.predict(x_test)

# Calculate Mean Squared Error (MSE)
mse = mean_squared_error(y_test, y_predict)
print(f"Mean Squared Error (RMSE): {mse:.2f}")

# Calculate R-squared (R2) Error
r2 = r2_score(y_test, y_predict)
print(f"R-squared (R2) Error: {r2:.2f}")
```

```
Mean Squared Error (RMSE): 0.05
R-squared (R2) Error: 0.98
```

1.3.1 Conclusion:

```
MSE of 0.05 indicates that model's predictions are very accurate & low errors.
R2 value of 0.98 suggests that model is highly effective at explaining & predicting the target
Decision Tree Regressor model is performing exceptionally well on your test data.
```

1.4 THANK YOU!!!

1.5 GitHub Link: